DATA

Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags	Consumer consent provided?	Submitted via	Date sent to company	Company response to consumer
2023-08-25	Credit reporting or other personal consumer re...	Credit reporting	Incorrect information on your report	Information belongs to someone else	NaN	NaN	EQUIFAX, INC.	FL	33009	NaN	NaN	Web	2023-08-25	Closed with explanation
2023-08-25	Credit reporting or other personal consumer re...	Credit reporting	Improper use of your report	Credit inquiries on your report that you don't...	NaN	NaN	EQUIFAX, INC.	MI	48234	NaN	NaN	Web	2023-08-25	Closed with explanation

```
[6]  data.shape

     (4051252, 18)
```

List of companies and the count of companies present in the data

```
[10] data['Company'].nunique()

     6923

[11] data['Company'].unique()

     array(['EQUIFAX, INC.', 'TRANSUNION INTERMEDIATE HOLDINGS, INC.',
            'Experian Information Solutions Inc.', ..., 'PREFERRED LOANS LLC',
            'Eckhoff & Massarelli, P.C.', 'Parkstone Mortgage, LLC'],
           dtype=object)
```

```
1s  data.describe()
```

	Complaint ID
count	4.051252e+06
mean	4.311083e+06
std	2.022199e+06
min	1.000000e+00
25%	2.867282e+06
50%	4.242992e+06
75%	6.130034e+06
max	7.525660e+06

```
data['Product'].unique()
```

```
array(['Credit reporting or other personal consumer reports',
       'Credit reporting, credit repair services, or other personal consumer reports',
       'Debt collection', 'Mortgage', 'Checking or savings account',
       'Credit card or prepaid card',
       'Payday loan, title loan, or personal loan', 'Student loan',
       'Vehicle loan or lease', 'Credit card',
       'Money transfer, virtual currency, or money service',
       'Debt or credit management', 'Bank account or service',
       'Payday loan, title loan, personal loan, or advance loan',
       'Credit reporting', 'Payday loan', 'Prepaid card', 'Consumer Loan',
       'Money transfers', 'Other financial service', 'Virtual currency'],
      dtype=object)
```

As stated the data needs to categorized into following four types which are ‘Credit reporting, credit repair services, or other personal consumer reports’,’Consumer Loan’,’Debt collection’,’Mortgage’ and filtered data accordingly

```
[16] if data is not None:
        # Specify the categories you want to keep in a list
        target_categories = ["Mortgage", "Consumer Loan", "Debt collection", "Credit reporting, credit repair services, or other personal consumer reports"]

        # Use the isin() method to create a boolean mask
        mask = data['Product'].isin(target_categories)

        # Use the mask to filter the DataFrame and keep only rows with the specified categories
        data1 = data[mask]
```

```
data1
```

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags	Consumer consent provided?	Submitted via	Date sent to company	Company response to consumer	res
2	2023-08-23	Credit reporting, credit repair services, or o...	Credit reporting	Problem with a credit reporting company's inve...	Their investigation did not fix an error on yo...	NaN	NaN	EQUIFAX, INC.	GA	30034	NaN	NaN	Web	2023-08-23	In progress	

The altered data shape is

```
[20] data1.shape
```

```
(3087188, 17)
```

To find the unique zip codes present for each state

```python
unique_records = set()
# Iterate through the DataFrame and add unique combinations to the set
for index, row in data1.iterrows():

        state = row['State']
        zipcode = row['ZIP code']
        unique_records.add((state, zipcode))

for state, zipcode in unique_records:
        print(f"State: {state}, ZIP Code: {zipcode}")
```

```
State: AE, ZIP Code: 09824
State: AE, ZIP Code: 09575
State: OH, ZIP Code: 45656
State: PA, ZIP Code: 15728
State: IA, ZIP Code: 50034
State: NY, ZIP Code: 14042
State: WY, ZIP Code: 255XX
State: MT, ZIP Code: 59414
State: NC, ZIP Code: 28633
State: ND, ZIP Code: 58501
State: NC, ZIP Code: 28344
State: WI, ZIP Code: 53186
State: FL, ZIP Code: 32807
State: TX, ZIP Code: 78584
State: VI, ZIP Code: 00841
State: NJ, ZIP Code: 07055
State: TX, ZIP Code: 79930
State: AL, ZIP Code: 36442
State: CA, ZIP Code: 94605
State: FL, ZIP Code: 33905
State: CT, ZIP Code: 06428
State: TN, ZIP Code: 37811
State: IN, ZIP Code: 47265
State: PA, ZIP Code: 18407
State: TX, ZIP Code: 75120
State: ID, ZIP Code: 83402
State: OH, ZIP Code: 45658
State: SC, ZIP Code: 29915
State: PA, ZIP Code: 16863
State: MA, ZIP Code: 01090
State: AR, ZIP Code: 85705
State: MA, ZIP Code: 01230
State: IL, ZIP Code: 62312
State: NJ, ZIP Code: 09302
State: KY, ZIP Code: 41360
```

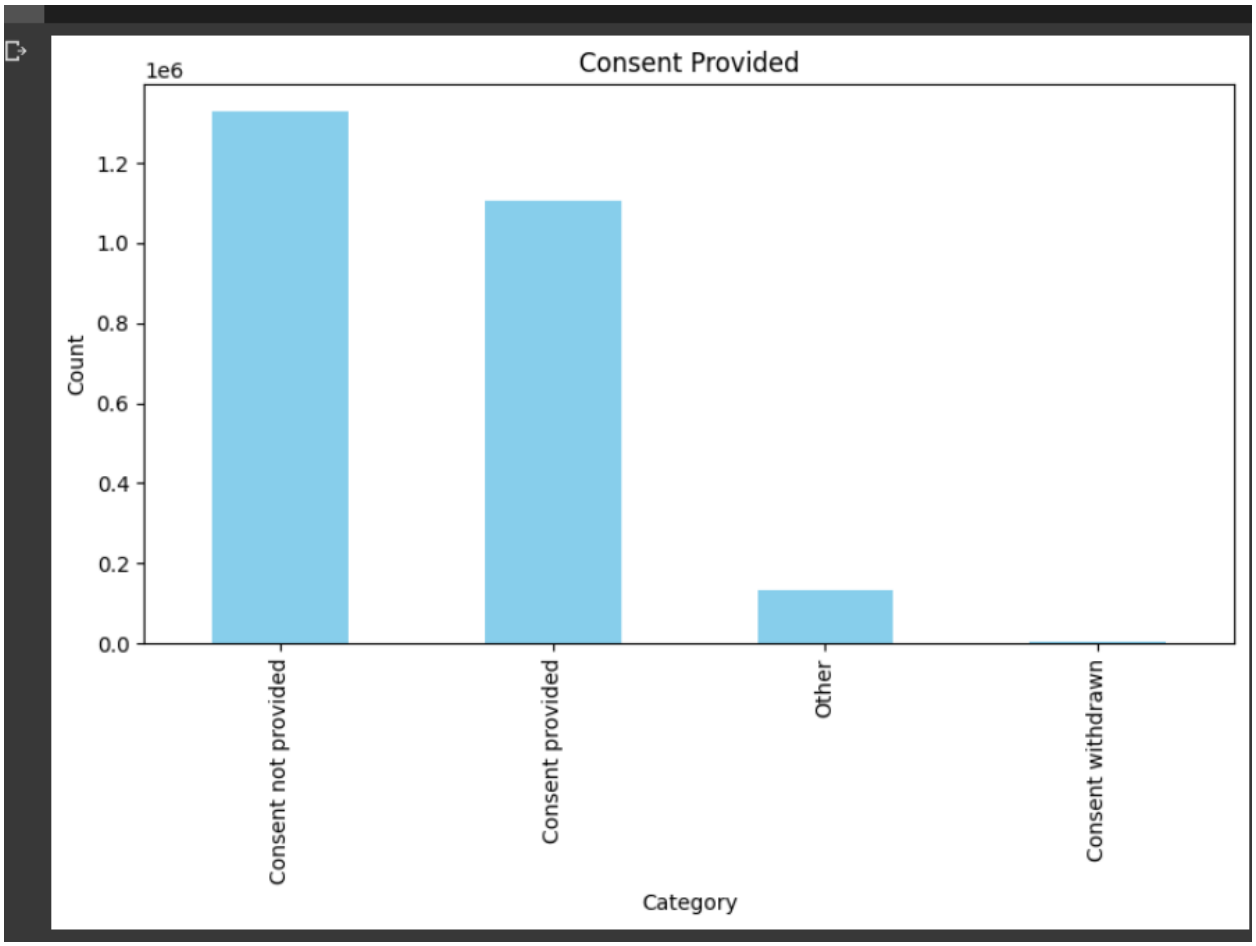
Also made an dictionary which has unique zip codes for each state

```python
unique_zipcodes_by_state = {}


unique_states = data1['State'].unique()
for state in unique_states:
        state_df = data1[(data1['State'] == state) & (data1['ZIP code'] != 'XXXXX')]
        unique_zipcodes = state_df['ZIP code'].unique()
        unique_zipcodes_by_state[state] = unique_zipcodes.tolist()

print(unique_zipcodes_by_state)
```

```
{'GA': ['30034', '30228', '30157', '31405', '30274', '30094', '30076', '30088', '30349', '30294', '30458', '30291', '30064', '30135', '30213', '30083', '30236', '30024
```

Categories available in **‘Consumer consent provided?’** column with the count

```
[72] data1['Consumer consent provided?'].unique()

     array([nan, 'Other', 'Consent provided', 'Consent not provided',
            'Consent withdrawn'], dtype=object)

     category_counts = data1['Consumer consent provided?'].value_counts()
     print(category_counts)

     Consent not provided    1329179
     Consent provided        1105054
     Other                    132845
     Consent withdrawn          5119
     Name: Consumer consent provided?, dtype: int64
```

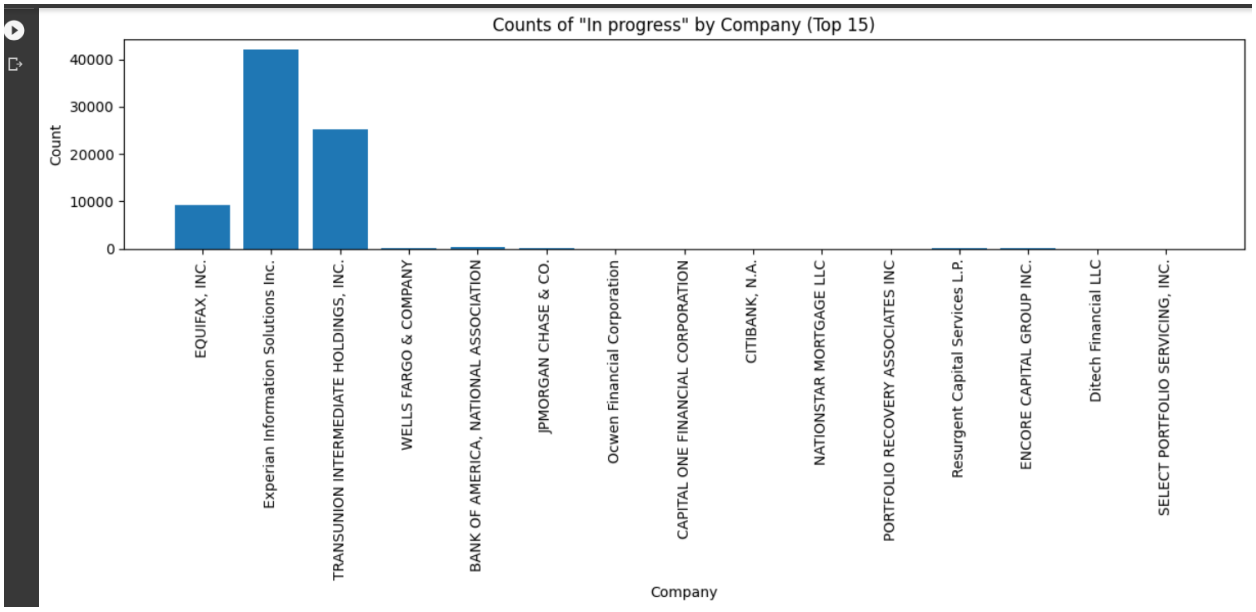


Company response to consumer with unique categories

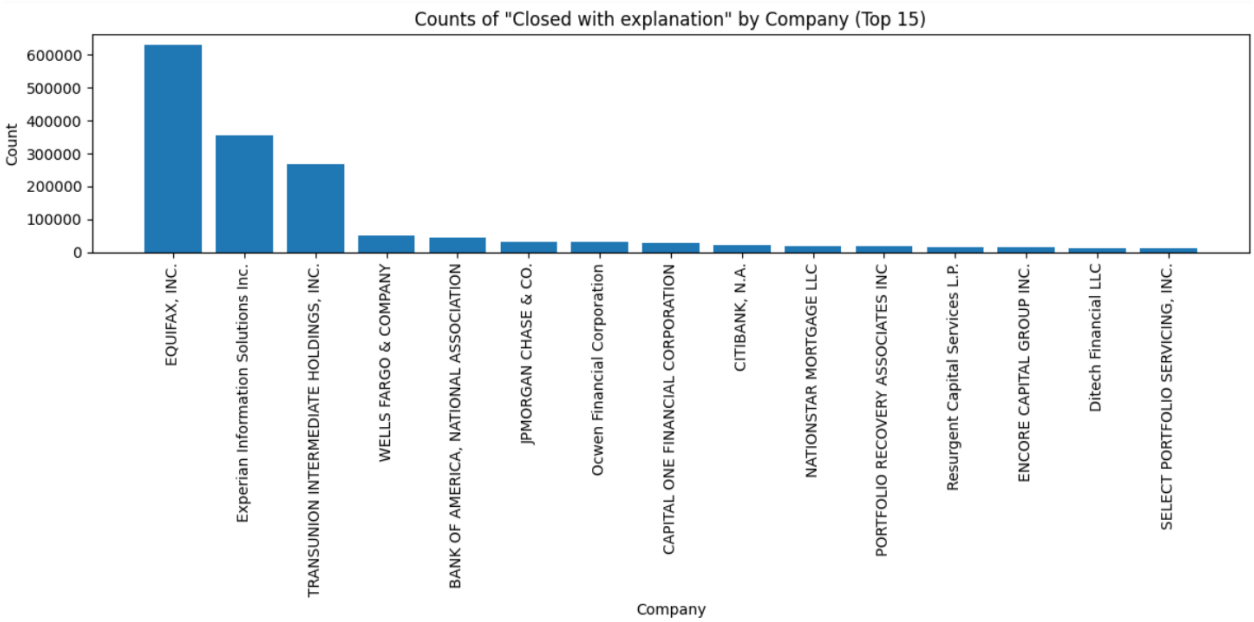
```
[66] data1['Company response to consumer'].unique()

array(['In progress', 'Closed with explanation',
       'Closed with non-monetary relief', 'Closed with monetary relief',
       'Closed without relief', 'Untimely response', 'Closed',
       'Closed with relief'], dtype=object)
```

Count of each category of **Company response to consumer** for every company

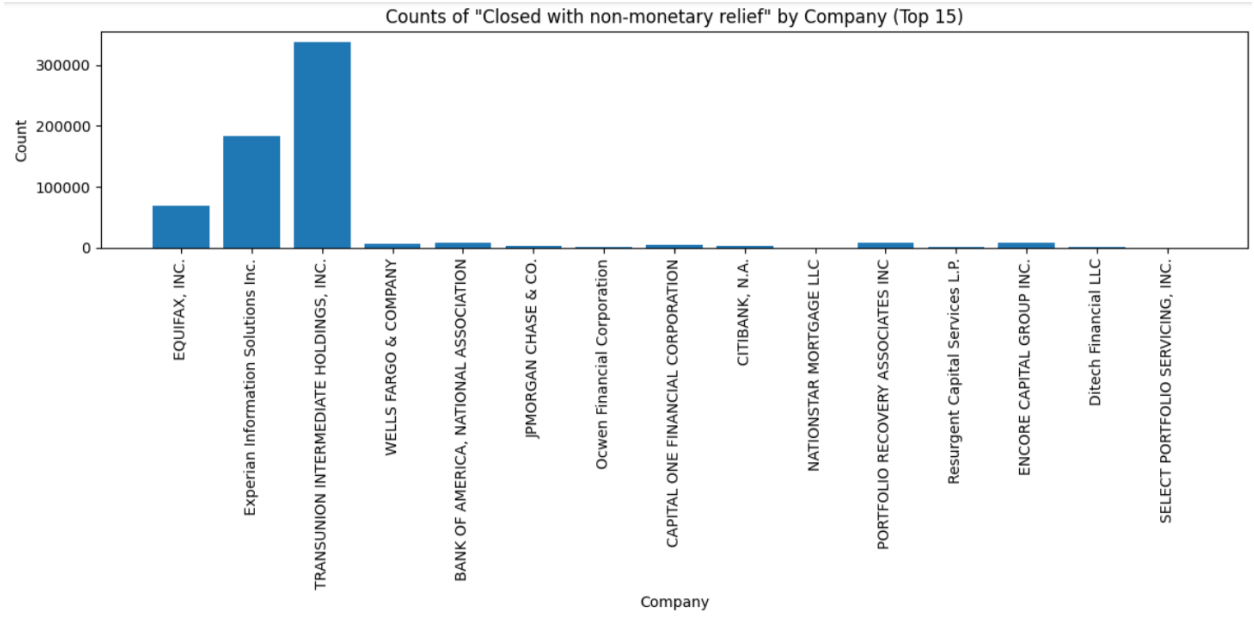
IN PROGRESS



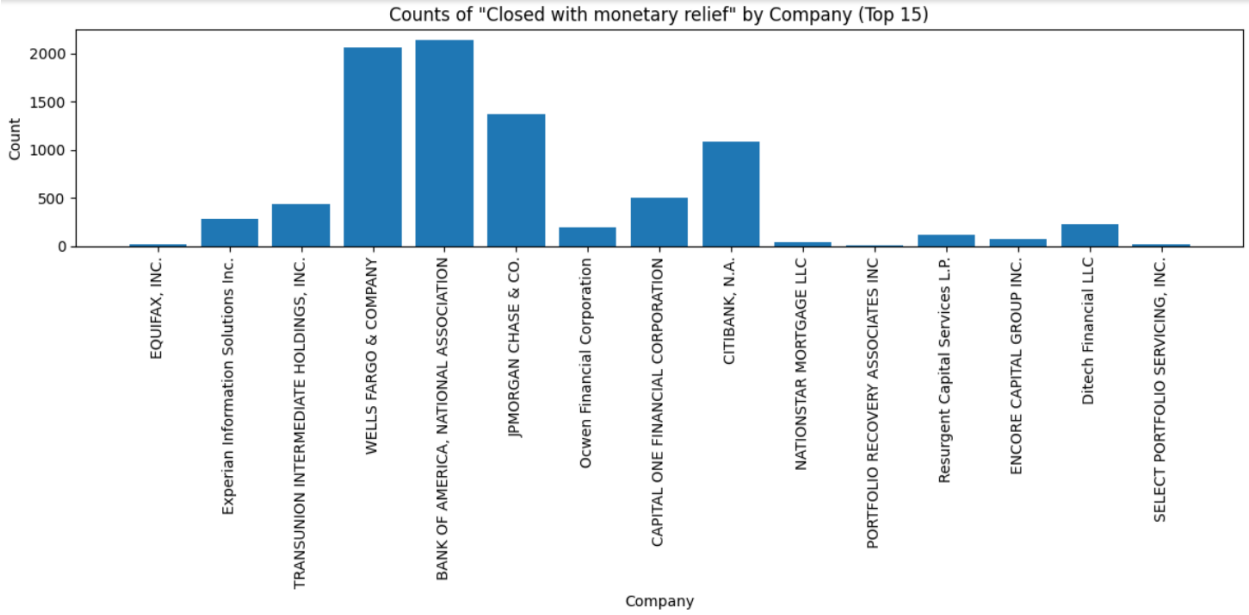
CLOSED WITH EXPLANATION



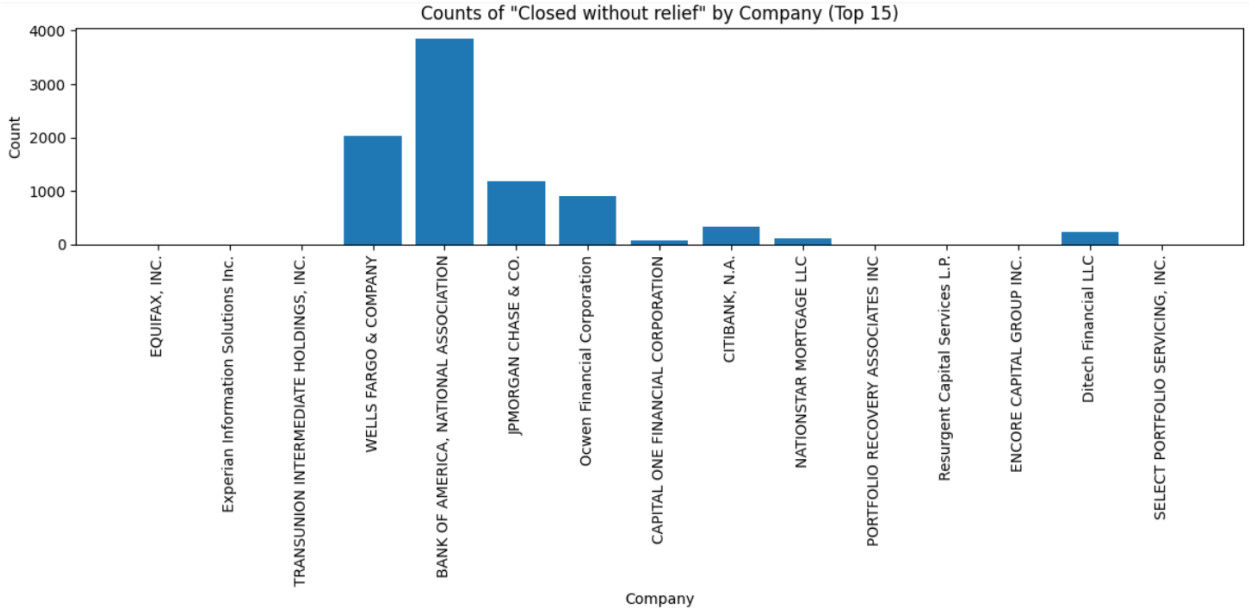
CLOSED WITH NON-MONETARY RELIEF



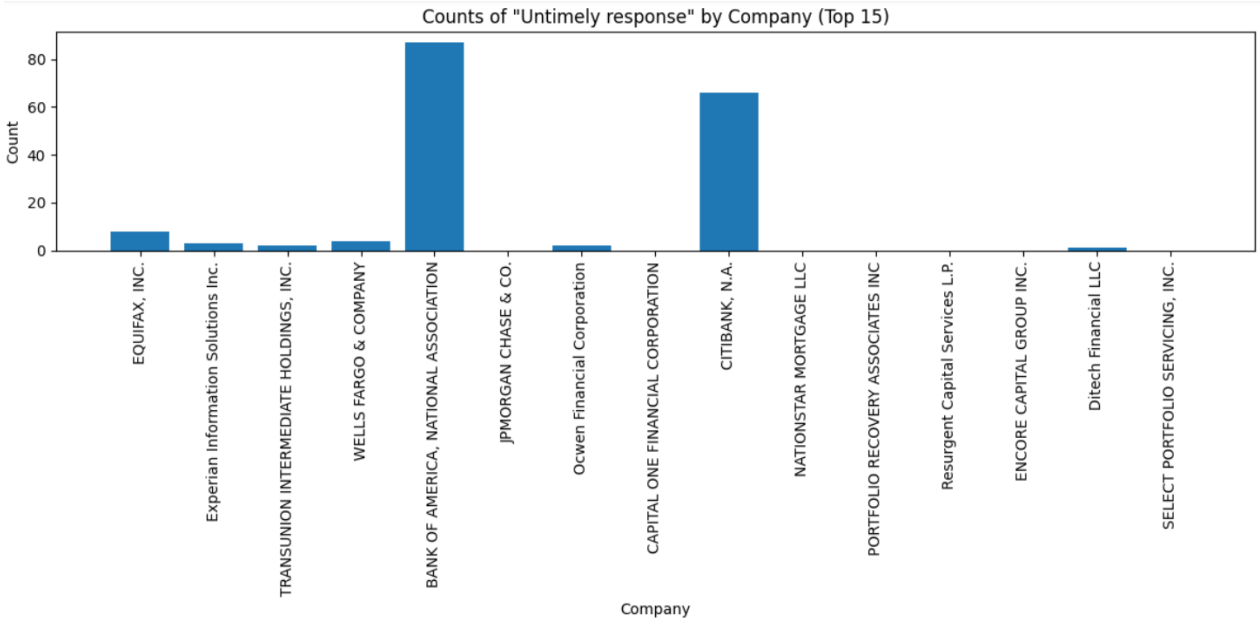
CLOSED WITH MONETARY RELIEF



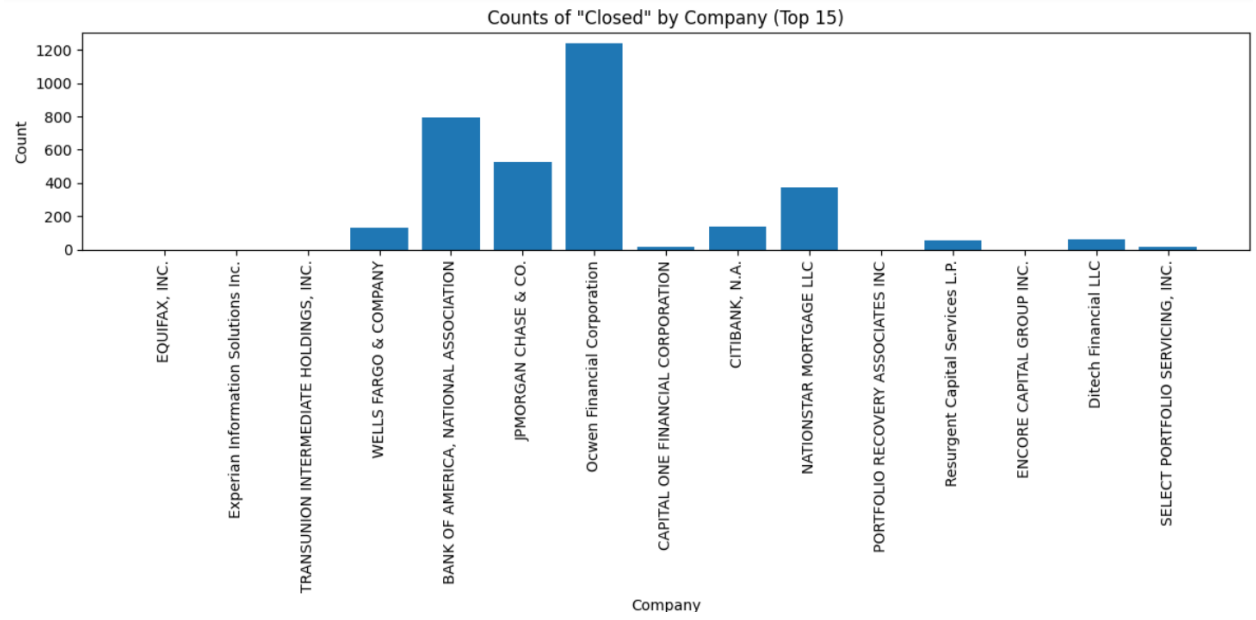
CLOSED WITHOUT RELIEF



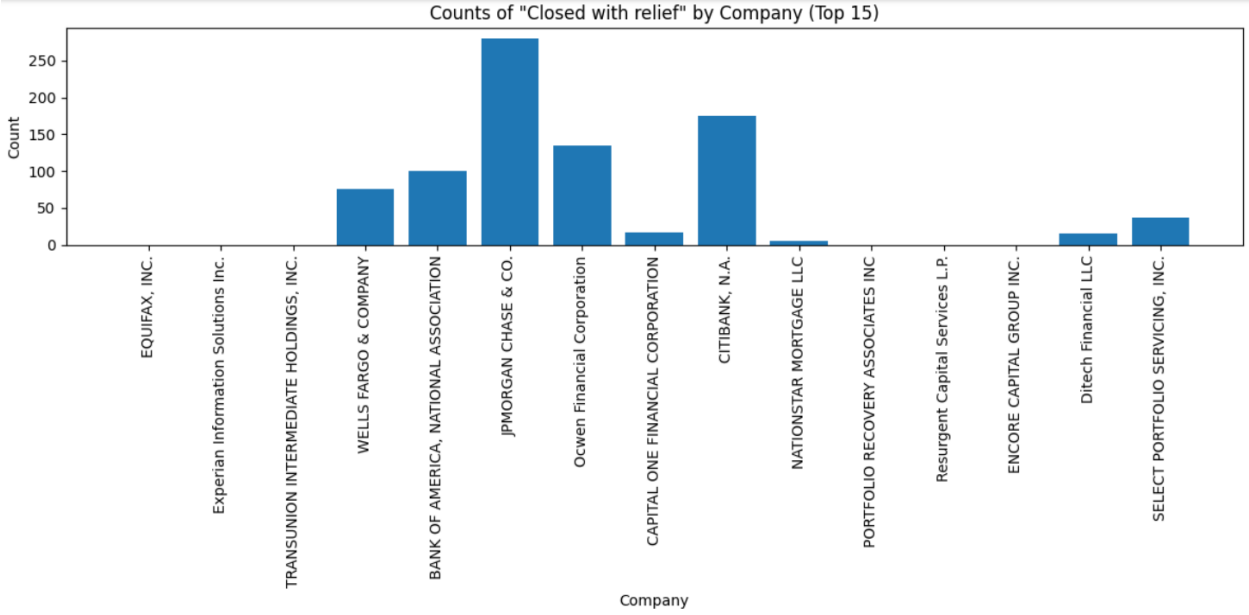
UNTIMELY RESPONSE



CLOSED



CLOSED WITH RELIEF



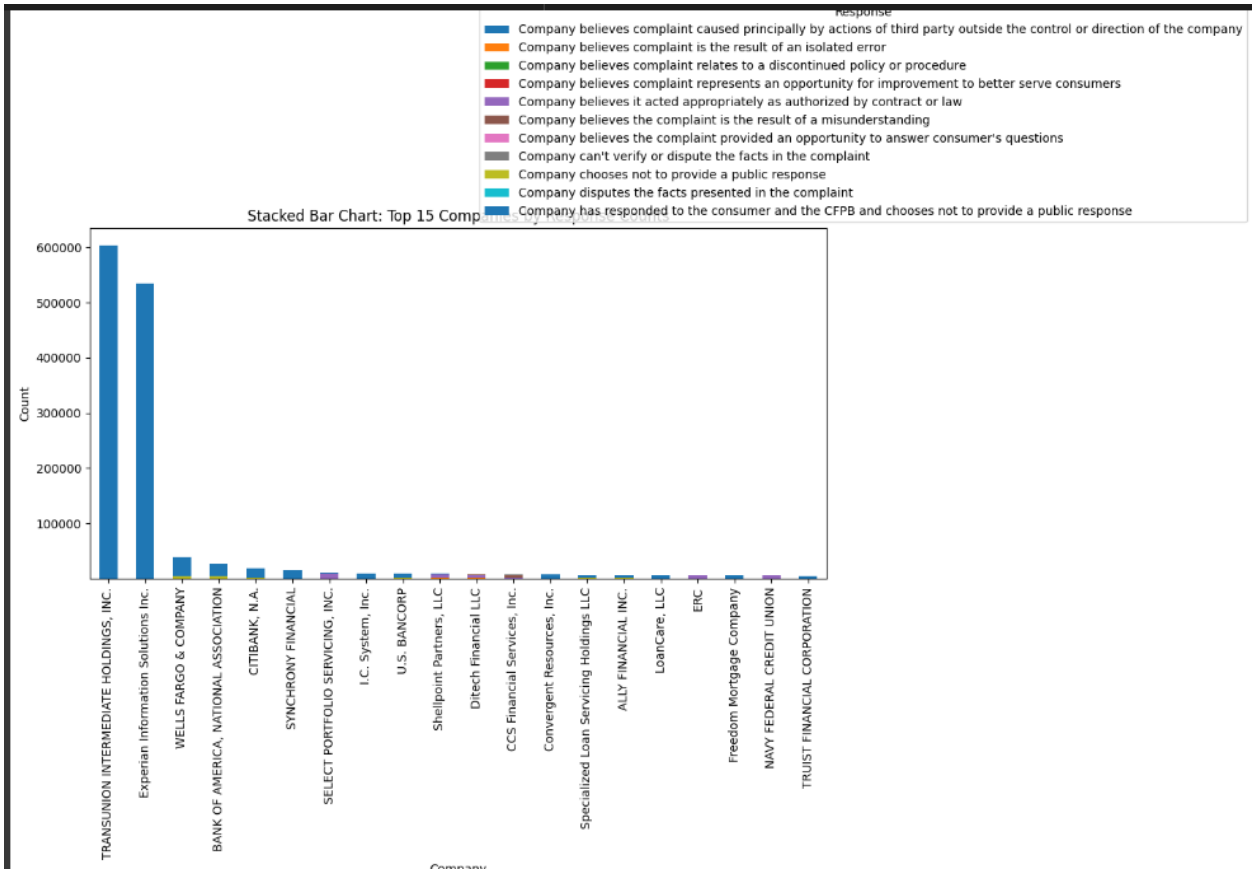
Individual **company public responses** with count

As observed **'Company has responded to the consumer and the CFPB and chooses not to provide a public response'** has highest count

```
[90] a=data1['Company public response'].value_counts()
     print(a)
     #out= data1.groupby(['Company', 'Company public response']).size().unstack(fill_value=0)
     #print(out)
```

```
Company has responded to the consumer and the CFPB and chooses not to provide a public response    1364761
Company believes it acted appropriately as authorized by contract or law                             97807
Company chooses not to provide a public response                                                     18521
Company believes the complaint is the result of a misunderstanding                                   11275
Company disputes the facts presented in the complaint                                                 8895
Company believes complaint caused principally by actions of third party outside the control or direction of the company    5947
Company believes complaint is the result of an isolated error                                         4617
Company can't verify or dispute the facts in the complaint                                            3860
Company believes complaint represents an opportunity for improvement to better serve consumers        3773
Company believes the complaint provided an opportunity to answer consumer's questions                 3012
Company believes complaint relates to a discontinued policy or procedure                                88
Name: Company public response, dtype: int64
```

Stacked bar representation of above variable



<u>Timely response</u> comparison by **<u>'State'</u>** and **<u>'Company'</u>**

The values in 'Yes' and 'No' columns determine which company gave the timely responses to its consumers and which did not.

```
timely_response_counts = data1['Timely response?'].value_counts()

    # Create a table with counts
state_response_counts = data1.groupby(['State', 'Timely response?']).size().unstack(fill_value=0)
print(state_response_counts)
```

```
Timely response?   No    Yes
State
AA                  0     38
AE                 10    742
AK                 36   2309
AL                562  65652
AP                  9    395
...               ...    ...
VT                 40   1525
WA                849  32305
WI                368  22659
WV                124   4093
WY                 59   1613

[63 rows x 2 columns]
```

```
timely_response_counts = data1['Timely response?'].value_counts()

    # Create a table with counts
company_response_counts = data1.groupby(['Company', 'Timely response?']).size().unstack(fill_value=0)
print(pd.DataFrame(company_response_counts))
```

```
Timely response?                          No  Yes
Company
(Former)Shapiro, Swertfeger & Hasty, LLP  12    0
1 Auto Finance, Inc.                       3    0
1 STOP MONEY CENTERS, LLC                  1    2
10 Cent Title Pawn Inc                     0    1
16 Hands LLC. dba Fiducius                 0    1
...                                       ..  ...
iQuantified Management Services, LLC       3   19
iReverse Home Loans, Corporation           0    1
snapfi, inc.                               0    2
snw investments                            0    2
 Lippman Recupero, LLC                     1    8

[6458 rows x 2 columns]
```

Companies that have ‘0’ yes count in a timely response which indicates that those companies do not give a timely response at all.The count of the companies those do not respond at all are 559 companies

```
companies_with_zero_yes = state_response_counts[state_response_counts['Yes'] == 0]
count_of_companies_with_zero_yes = len(companies_with_zero_yes)

if count_of_companies_with_zero_yes > 0:
    print("Companies with 0 'yes' count:")
    print(companies_with_zero_yes.index.tolist())
    print("Count of companies with 0 'yes' count:", count_of_companies_with_zero_yes)
else:
    print("No companies have 0 'yes' count.")
```

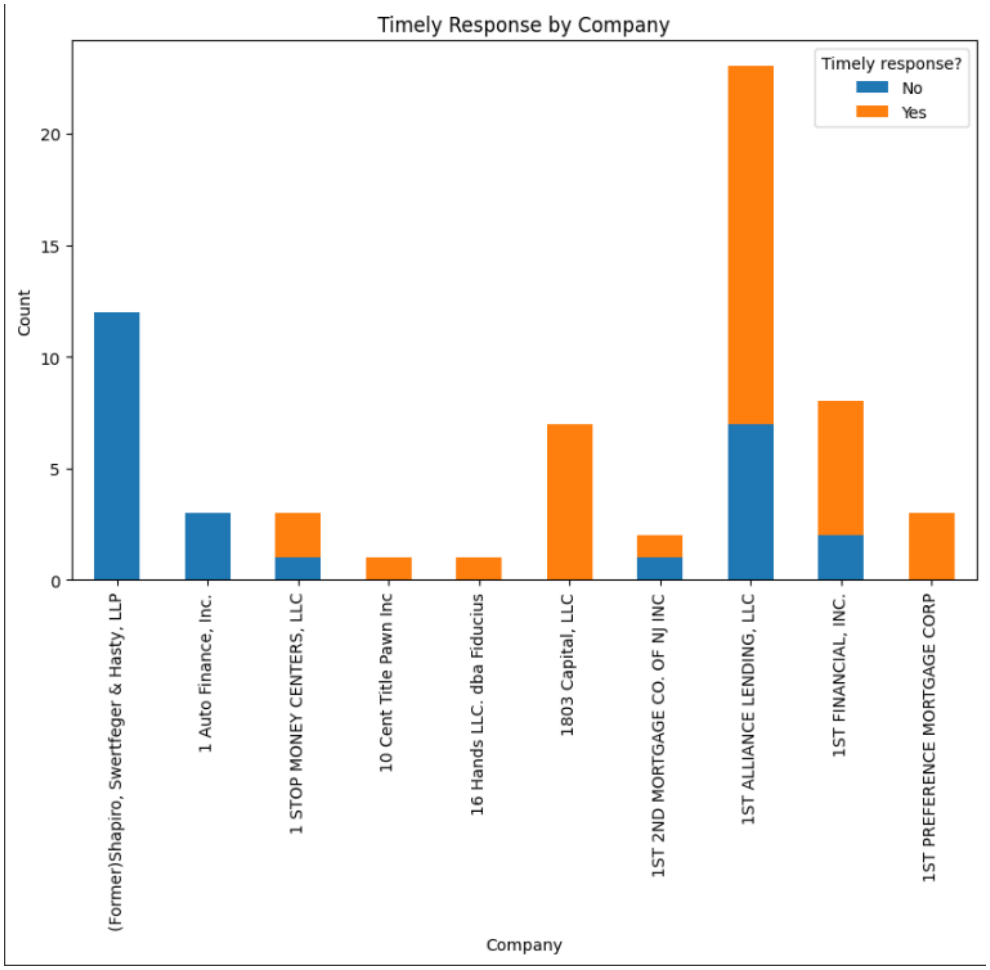
```
Companies with 0 'yes' count:
['(Former)Shapiro, Swertfeger & Hasty, LLP', '1 Auto Finance, Inc.', '360 Mortgage Inc.', 'A & O Recovery Solutions, LLC', 'A Credits Works', 'A.R.C. Accounts Recovery (U.S.A.) Corporation
Count of companies with 0 'yes' count: 559
```

Similarly for zero ‘No’ which means the company that definitely gives timely response to its consumers.

```
timely_response_counts = data1['Timely response?'].value_counts()
company_response_counts = data1.groupby(['Company', 'Timely response?']).size().unstack(fill_value=0)

companies_with_zero_no = company_response_counts[company_response_counts['No'] == 0]
count_of_companies_with_zero_no = len(companies_with_zero_no)


if count_of_companies_with_zero_no > 0:
    print("Companies with 0 'No' count:")
    print(companies_with_zero_no.index.tolist())
    print("Count of companies with 0 'No' count:", count_of_companies_with_zero_no)
else:
    print("No companies have 0 'No' count.")
```

```
Companies with 0 'No' count:
['10 Cent Title Pawn Inc', '16 Hands LLC. dba Fiducius', '1803 Capital, LLC', '1ST PREFERENCE MORTGAGE CORP', '1ST RESULTS BILLINGS & COLLECTIONS, INC.', '1st Capital Finance of South Carc
Count of companies with 0 'No' count: 2856
```

There are a total of 2856 companies that gives timely response.

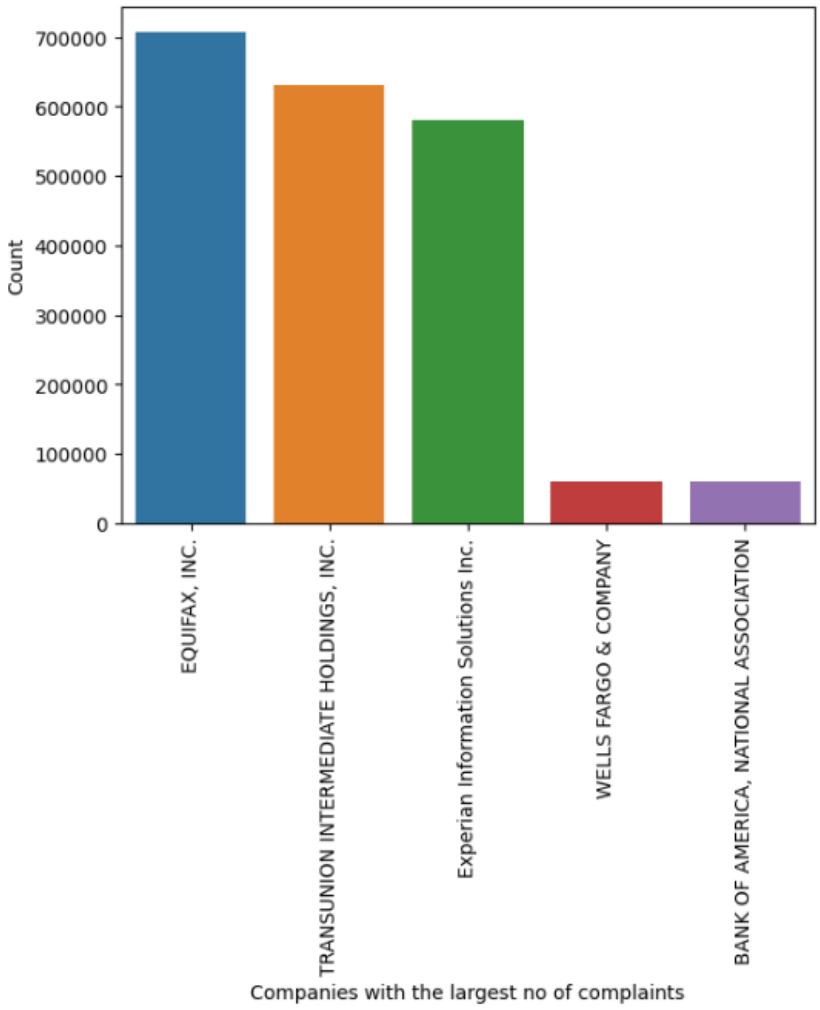


Some of the companies that respond timely are **10 Cent Title Pawn Inc**,**16 Hands LLC. dba Fiducius, 1803 Capital, LLC** and so on.

Count of entries to each company

```
companies = data1.groupby('Company').Company.count().sort_values(ascending=False)
print(len(companies))
companies.head(15)
```

```
6458
Company
EQUIFAX, INC.                              707879
TRANSUNION INTERMEDIATE HOLDINGS, INC.     630851
Experian Information Solutions Inc.        581048
WELLS FARGO & COMPANY                       60496
BANK OF AMERICA, NATIONAL ASSOCIATION       59878
JPMORGAN CHASE & CO.                        38481
Ocwen Financial Corporation                 33928
CAPITAL ONE FINANCIAL CORPORATION           33916
CITIBANK, N.A.                              27319
PORTFOLIO RECOVERY ASSOCIATES INC           24684
ENCORE CAPITAL GROUP INC.                   22071
NATIONSTAR MORTGAGE LLC                     19241
SYNCHRONY FINANCIAL                         17562
Resurgent Capital Services L.P.             15628
Ditech Financial LLC                        14849
Name: Company, dtype: int64
```



COMPANY VS PRODUCT

```
company_a_data = data1[data1['Company'] == 'EQUIFAX, INC.']


product_counts = company_a_data['Product'].value_counts()


if not product_counts.empty:
    highest_count_product = product_counts.idxmax()
    highest_count = product_counts.max()

    print(f"For company 'EQUIFAX, INC.', the highest count product is '{highest_count_product}' with {highest_count} occurrences.")
else:
    print("No data found for company 'EQUIFAX, INC.'.")
```

```
For company 'EQUIFAX, INC.', the highest count product is 'Credit reporting, credit repair services, or other personal consumer reports' with 699023 occurrences.
```

```
For company 'snw investments', the highest count product is 'Mortgage' with 2 occurrences.
```

```
For company 'SYNCHRONY FINANCIAL', the highest count product is 'Credit reporting, credit repair services, or other personal consumer reports' with 9213 occurrences.
```

This gives the largest no of complaints that each company has on the ‘Product’.

List of **'Issues'**

```
issue_counts = data1['Issue'].value_counts()
issue_counts
```

```
Incorrect information on your report                                    1020461
Problem with a credit reporting company's investigation into an existing problem     579950
Improper use of your report                                              511097
Attempts to collect debt not owed                                        180953
Loan modification,collection,foreclosure                                 112306
                                                                            ...
Struggling to pay your loan                                                   1
Closing your account                                                          1
Problem with the payoff process at the end of the loan                        1
Fees or interest                                                              1
Getting a line of credit                                                      1
Name: Issue, Length: 70, dtype: int64
```

List of **'Sub-issues'**

```
data1['Sub-issue'].value_counts()
```

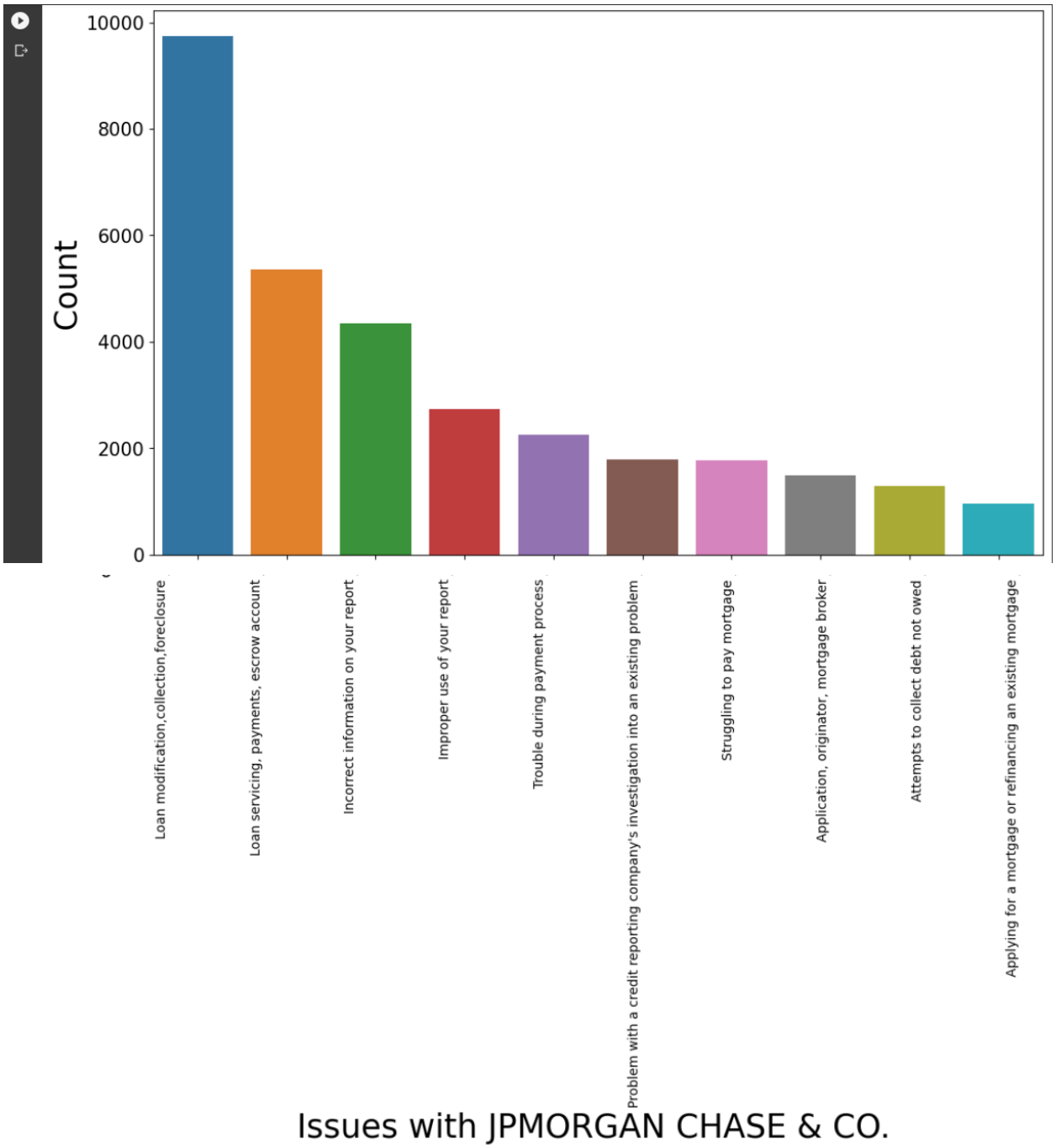
```
Information belongs to someone else                          675880
Reporting company used your report improperly                342256
Their investigation did not fix an error on your report      330222
Credit inquiries on your report that you don't recognize     165554
Investigation took more than 30 days                         130542
                                                                ...
Problem during payment process                                    1
Can't close your account                                          1
Denied request to lower payments                                  1
Problem with fraud alerts or security freezes                     1
Application denied                                                1
Name: Sub-issue, Length: 120, dtype: int64
```

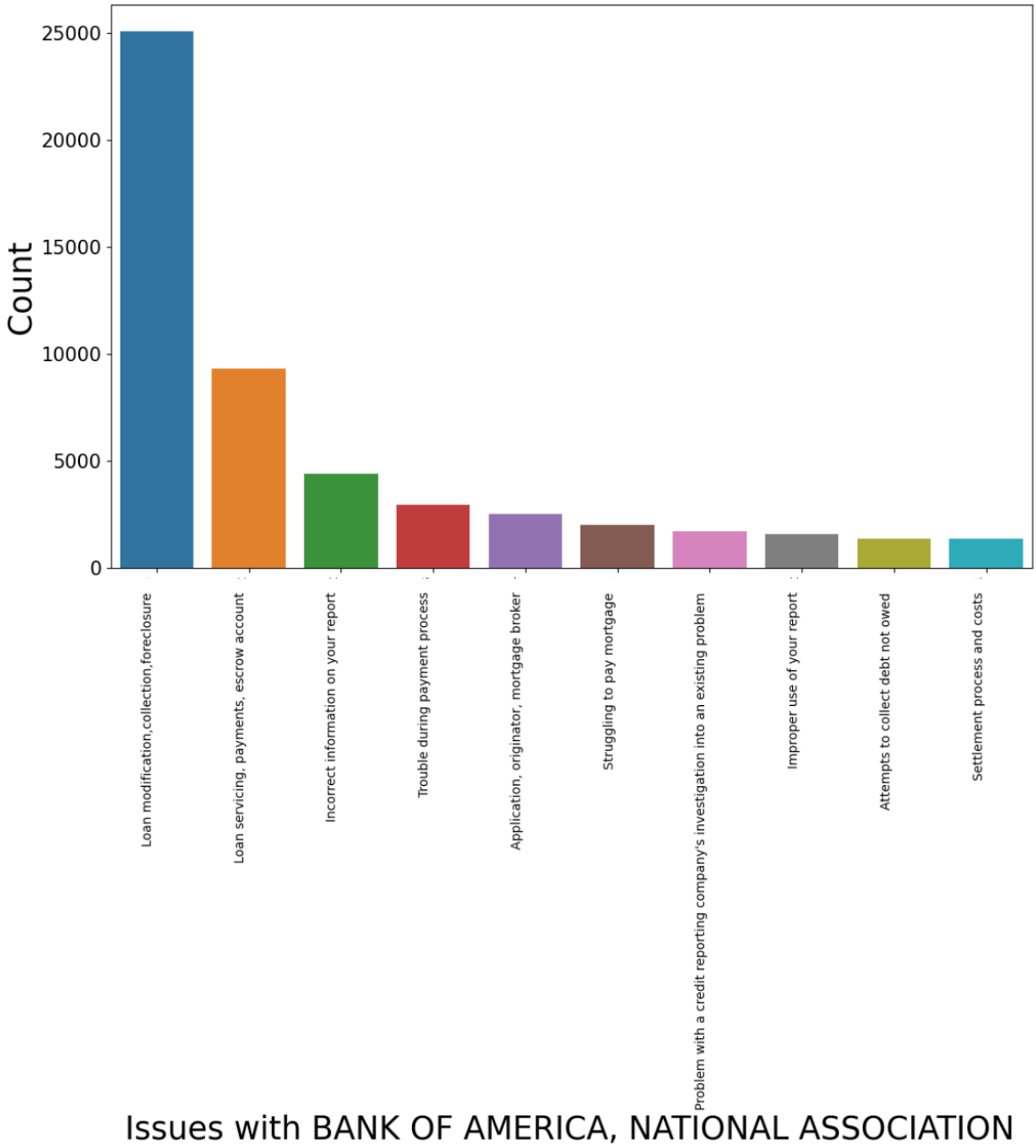
No of consumers disputed

```
[35] data1['Consumer disputed?'].value_counts()
```

```
No     320439
Yes     83774
Name: Consumer disputed?, dtype: int64
```

Graphical representation of Sub-issue count to each company

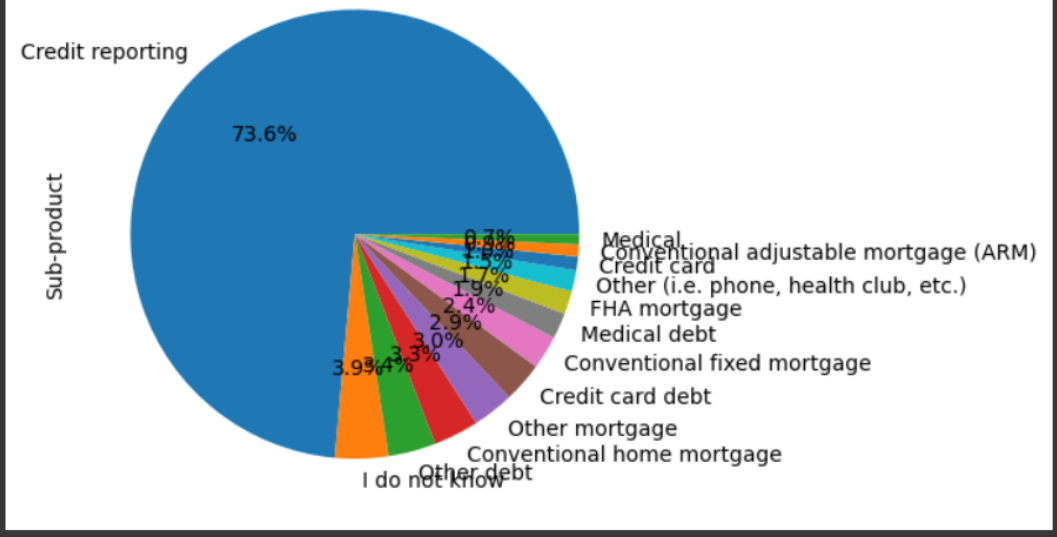



Count
25000
20000
15000
10000
5000
0
Loan modification,collection,foreclosure
Loan servicing, payments, escrow account
Incorrect information on your report
Trouble during payment process
Application, originator, mortgage broker
Struggling to pay mortgage
Problem with a credit reporting company's investigation into an existing problem
Improper use of your report
Attempts to collect debt not owed
Settlement process and costs
Issues with BANK OF AMERICA, NATIONAL ASSOCIATION

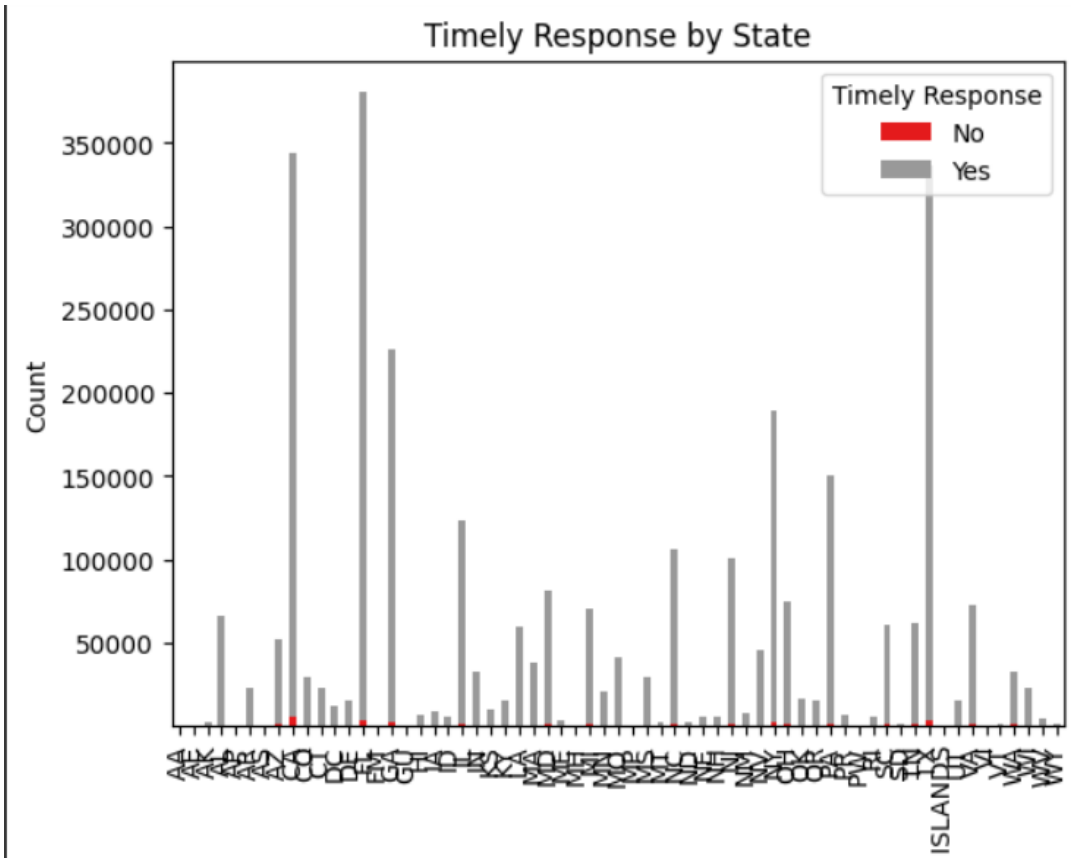
Percentage of each **‘Sub-product’**

```
data1["Sub-product"].value_counts().head(13).plot(kind="pie", autopct="%.1f%%")
```

```
<Axes: ylabel='Sub-product'>
```



Timely response VS State



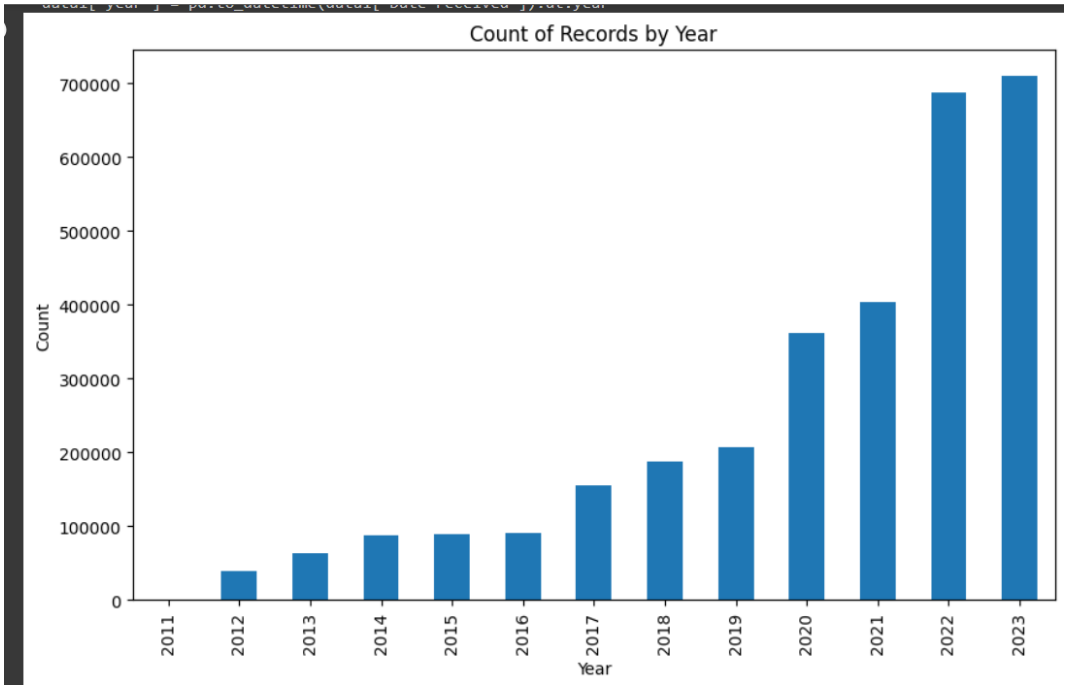
```
state_response_counts = data1.groupby(['State', 'Timely response?']).size().unstack(fill_value=0)
state_response_counts['Percentage Yes'] = (state_response_counts['Yes'] / state_response_counts.sum(axis=1)) * 100

    # Find the state with the lowest percentage of 'Yes' responses
least_responsive_state = state_response_counts['Percentage Yes'].idxmin()
lowest_percentage = state_response_counts['Percentage Yes'].min()
print("the least responsive state is:"+least_responsive_state)
print(lowest_percentage)
```

```
the least responsive state is:MH
94.44444444444444
```

```
[47] state_response_counts = data1.groupby(['State', 'Timely response?']).size().unstack(fill_value=0)
     state_response_counts['Percentage Yes'] = (state_response_counts['Yes'] / state_response_counts.sum(axis=1)) * 100

         # Find the state with the highest percentage of 'Yes' responses
     most_active_state = state_response_counts['Percentage Yes'].idxmax()
     highest_percentage = state_response_counts['Percentage Yes'].max()
     print("the highest responsive state is:"+most_active_state)
     print(highest_percentage)
```

```
the highest responsive state is:AA
100.0
```

The above list the states and response percentage with least and highest response. The least responsive state is '**MH**' and highest responsive state is '**AA**'.

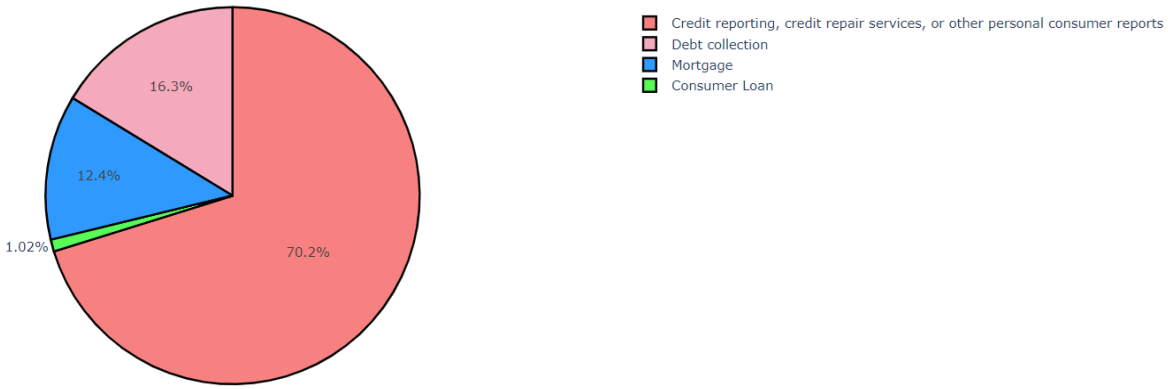


The above graph identifies that the complaints increase year by year.

Count of each target product

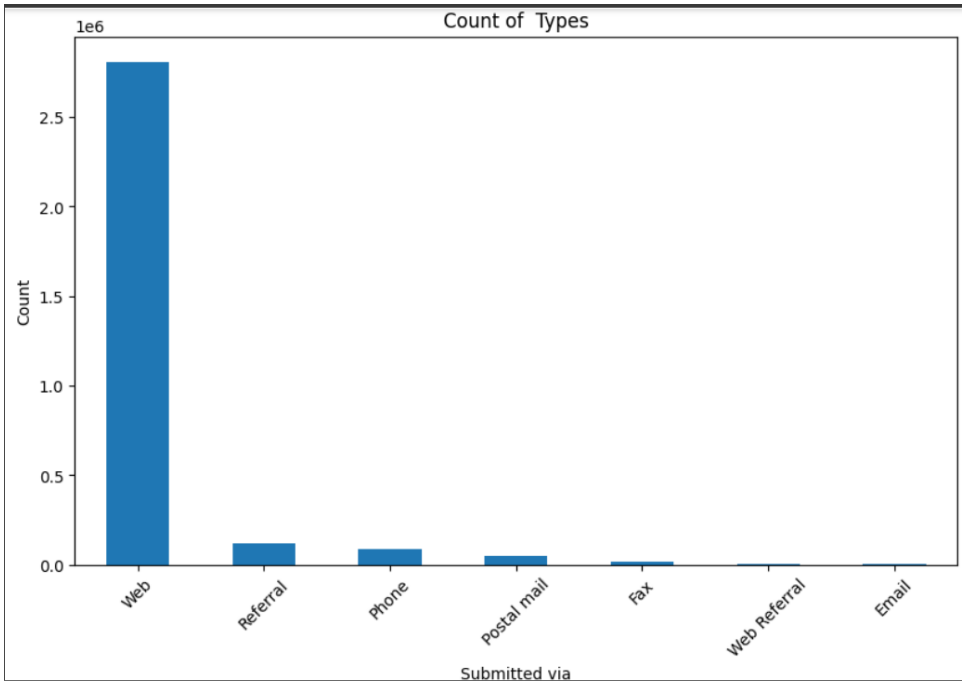
```
target_product = 'Credit reporting, credit repair services, or other personal consumer reports'
filtered_df = data1[data1['Product'] == target_product]

sub_product_counts = filtered_df['Sub-product'].value_counts()
print(sub_product_counts)
```

```
Credit reporting                 2146468
Other personal consumer report     16045
Credit repair services              5052
Conventional home mortgage             1
Name: Sub-product, dtype: int64
```



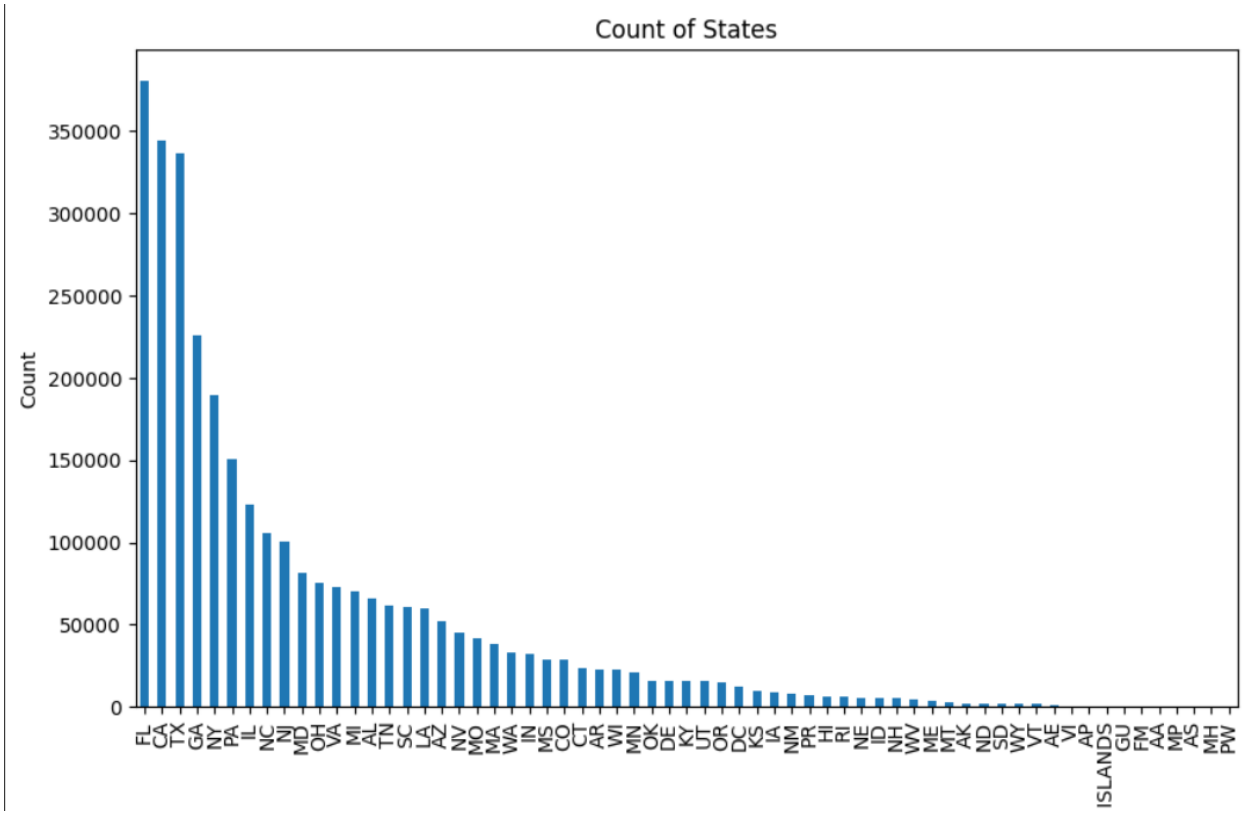
The visualization tells that more than half the data falls under **'Credit reporting, credit repair services, or other personal consumer reports'** category.

SUBMITTED VIA



The type of source that consumers filed the complaint. As plotted most of the consumers prefer ‘Web’.

State with the count of complaints



NAIVE BAYES CLASSIFIER

Accuracy: 0.89

	precision	recall	f1-score	support
0	0.53	0.23	0.32	1868
1	0.92	0.93	0.93	154041
2	0.79	0.72	0.75	43495
3	0.82	0.95	0.88	21492
accuracy			0.89	220896
macro avg	0.77	0.71	0.72	220896
weighted avg	0.88	0.89	0.88	220896

Stochastic Gradient Descent Classifier

```
[57] sgdc.score(X_train, y_train)

     0.9145942548048007
```

Prediction for SGDC

```
from sklearn.metrics import accuracy_score

y_pred = sgdc.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

new_text_samples = df1['Consumer complaint narrative'].head(1)
new_text_samples_scaled = cv.transform(new_text_samples)

new_text_predictions = sgdc.predict(new_text_samples_scaled)

for i, text_sample in enumerate(new_text_samples):
    print(f"Text: {text_sample}")
    print(f"Predicted Label: {new_text_predictions[i]}\n")
```

```
Accuracy: 0.91
Text: I am a victim of identity theft please remove this fraud inquiry from my credit report.
Predicted Label: 1
```

The below complaint is used for testing and the ‘Product’ prediction is correct.

```
df1
```

	Product	Consumer complaint narrative
14	1	I am a victim of identity theft please remove ...